Logical Characterizations of Truly Concurrent Bisimulation

S. Pinchinat[†,‡], F. Laroussinie[‡] and Ph. Schnoebelen[‡]

† School of Cognitive and Computing Sciences,
University of Sussex - Falmer, Brighton B1 9QH - UK
E-mail: sophiep@cogs.susx.ac.uk, Fax: +44 273 678188

‡ LIFIA-IMAG - 46 Av. Félix Viallet
38031 Grenoble Cedex - FRANCE
E-mail: phs,fl@imag.fr, Fax: +33 76574602

Abstract

We present two modal logics, interpreted over prime event structures without silent moves, that characterize history-preserving bisimulation. That is, two event structures satisfy the same formulae iff they are history-preserving bisimilar. These logics use past-tense modalities and pomsets observations. Though quite different in form and spirit, the two logics turn out to have the same expressive power, in the sense that there exists a two-way translation between them.

Keywords: Semantics of concurrency, modal logics for true concurrency, expressivity of modal logics.

Introduction

During the past decade, partial order semantics have been studied in depth as a promising approach to the semantic aspects of reactive systems [Gra81, GR83, Pra86]. In particular, causality-based models enable one to formalize interesting features of systems' behaviors for which concurrency is no longer identified with sequential non-determinism as it is in interleaving models.

As it has been pointed out [CMP87, Pra86], partial order semantic equivalences may be used to define abstract notions of behaviors in a way which is independent of the level of atomicity of the actions performed by the system. One such behavioral equivalence is *History Preserving Bisimulation*, simply called *hp-bisimulation* in this article.

Hp-bisimulation plays an important rôle in the literature. It was originally proposed by [RT88] for Petri nets as *BS-bisimulation*, and has also been considered in [BDKP91] under the name of *Fully Concurrent Bisimulation*. [GG89] adapted it to event structures, the basic models for studying the interplay of branching-time and causality, and proved that hp-bisimulation is preserved by refinement of actions.

Its many desirable theoretical properties have led to an in-depth study of hp-bisimulation. For example, [Vog91] established its decidability over finite safe Petri nets by reformulating hp-bisimulation in terms of *OM-bisimulation*. Several other semantic characterizations exist, for example [Vaa89, DDM89, Che92].

However, hp-bisimulation still lacks a characterization in terms of temporal (and more generally, modal) logics. Often such characterizations are a powerful tool from both theoretical and practical viewpoints. Here the classical result (from [HM80]) is the modal characterization of strong interleaving bisimulation by the HML modal logic. This result argued for bisimulation as a natural behavioral equivalence and illustrated some of the concepts behind the bisimulation approach. It also showed that HML can be used as a language in which to explain why two systems are not semantically equal [Hil87].

Since this fundamental result, many other behavioral equivalences have been characterized through modal logics, see [Mil81, BR83, HS85, Pnu85, DF90, DV90, LPS93] among others. Most of these results assume an interleaving setting. Indeed, in a causality-oriented setting, few modal characterizations exist (but see [GKP92]), and surprisingly we are aware of none for hp-bisimulation. This is certainly partly due to the fact that hp-bisimulation is not defined in term of states like most variants of bisimulation (e.g. pomset-bisimulation [BC87]), but in term of states with an history.

In this article, we propose two different modal logics, called L_{bf} and L_P, characterizing hp-bisimulation. Our logics are interpreted over event structures which are the most natural framework for such a study. Clearly, modal characterizations of hp-bisimulation can readily be obtained using the causal tree framework [DD89], since hp-bisimulation coincides with strong bisimulation over causal trees [Vaa89]. However, reasoning in terms of causal trees is less natural than using pomsets as local observable actions in event structures. Indeed, we propose logics based on pomset modalities. Moreover, to attain the distinguishing power of hp-bisimulation, we introduce past tense modalities in the logics. L_{bf} extends HML with pomset (instead of just action) labels, and with backward modalities. The second logic, L_P, has a CTL^* flavor (with past-time) and was inspired from the proposal in [DF90].

Beyond this, another issue we want to raise in this article is the comparative expressivity of modal logics for concurrent systems, and herein lies the main contribution of this article. Just stating that L_{bf} and L_P characterize hp-bisimulation does not answer all questions. At first sight, L_{bf} seems simpler than L_P. It only has basic modalities and a simple way of combining them, while L_P allow more complex statements over a selected future : indeed L_{bf} is easily encoded into a simple fragment of L_P. The second part of this article is devoted to establish a two-way translation between L_P and L_{bf}. That such a translation exists is quite surprising. The translation from L_P to L_{bf} is not trivial and uses methods developed in [LPS93]. Because of its constructive nature, our translation result has practical applications. For example, it shows that one can use the simpler L_{bf} logic to implement a verifier for truly concurrent systems, while retaining the more natural and intuitively more expressive L_P logic for the user interface, where specifications are written.

Beyond these applications, we would like to stress the novelty of our approach. In this sense, this article is a continuation of [LPS93] where we claimed that looking at expressivity issues for modal logics of reactive systems would be very fruitful. There we produced non-trivial translations between modal logics for interleaving semantics with silent moves. Now we explore truly concurrent semantics, where equally interesting results are available. Clearly these questions deserve further study, and we hope our contribution will motivate other researchers into looking at it.

The article is organized as follows : Section 1 presents the models, namely (prime) labeled events structures without silent actions. Section 2 introduces *mixed-ordering equivalence*, one possible way of defining hp-bisimulation. Both logics L_{bf} and L_P are presented in (resp.) Sections 3 and 4 where we prove they characterize mixed-ordering equivalence. Finally, Sections 5 and 6 are devoted to expressivity issues.

1 Labeled Event Structures

Event structures are well-known "truly concurrent" models of computational systems. They have been introduced by Nielsen, Plotkin and Winskel. See [Win89] for motivations and a complete technical exposition.

The main idea behind event structures is to view distributed computations as action occurrences, called *events*, together with a notion of causal dependency between events (which is reasonably formalized via a partial order). A labeling function records which action an event corresponds to. Moreover, in order to model non-determinism, there is a notion of mutually incompatible events.

1.1 Structure

In the following, we assume a given (finite) set $Act = \{a, b, c, \ldots\}$ of action names.

Definition 1.1 *A* Labeled Event Structure (LES *for short) is a structure* $\mathcal{E} = (E_{\mathcal{E}}, \leq_{\mathcal{E}}, \#_{\mathcal{E}}, l_{\mathcal{E}})$ *consisting of*

1. *a set* $E_{\mathcal{E}}$ *of* events*,*
2. *a mapping* $l_{\mathcal{E}} : E_{\mathcal{E}} \mapsto Act$*, called the* labeling function*,*
3. *a (reflexive) partial order* $\leq_{\mathcal{E}} \subseteq E_{\mathcal{E}} \times E_{\mathcal{E}}$*, called the* causal dependency relation*, and*
4. *a symmetric irreflexive relation* $\#_{\mathcal{E}} \subseteq E_{\mathcal{E}} \times E_{\mathcal{E}}$*, called the* conflict relation*,*

satisfying

1. *the* finiteness of causes assumption *: for all* $e \in E_{\mathcal{E}}$*, the set* $\{e' \mid e' \leq_{\mathcal{E}} e\}$ *is finite, and*
2. *the* conflict hereditary property *: for all* $e, e', e'' \in E_{\mathcal{E}}$*,* $e \#_{\mathcal{E}} e' \leq_{\mathcal{E}} e''$ *implies* $e \#_{\mathcal{E}} e''$*.*

In this article, we only consider finite event structures [1] ($\mathcal{E}$ is *finite* if $E_{\mathcal{E}}$ is).

We let **LES** $= \{\mathcal{E}, \mathcal{F}, \ldots\}$ denote the class of all event structures (over Act). Subscripts "$\mathcal{E}$" in $E_{\mathcal{E}}$, $\leq_{\mathcal{E}}$, etc, will be omited when they are clear from the context.

In graphical representation of LES, the causal dependency relation is represented by oriented arcs, omitting those derivable by transitivity. Similarly, only immediate conflicts, not inherited ones, are

[1] This is only a technical simplification : as they are stated, our main results apply to all event structures having finite branching, and could be extended to all event structures, without any restriction, by slightly modifying some definitions.

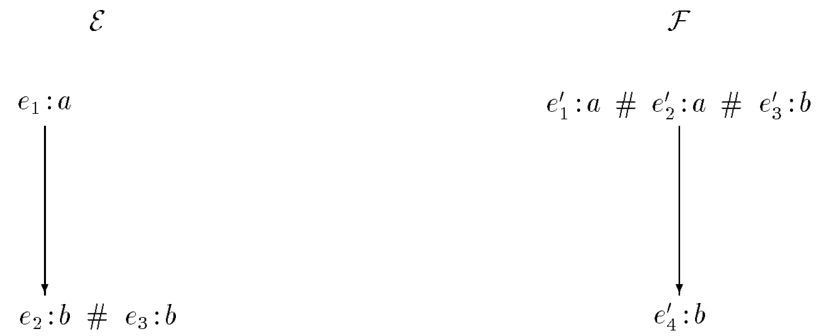

Figure 1: Two labeled event structures.

indicated (e.g. $e'_1 \#_{\mathcal{F}} e'_4$ is implicit in Figure 1). For convenience, we write down both the events and their labels as in $e : a$.

Usually, one does wish to abstract from the specific names of the events in an event structure : only the underlying structure matters. Formally, event structures are considered up to isomorphism :

Definition 1.2 *An* isomorphism *between $\mathcal{E}$ and $\mathcal{F}$ is a bijective mapping $f : E_{\mathcal{E}} \mapsto E_{\mathcal{F}}$ which preserves the relations of causality and conflict and the labeling functions.*

1.2 States and behaviors

The computational states in LES's, called *configurations*, are represented by sets of events : the events that have occurred so far. Formally,

Definition 1.3 *A* configuration *of $\mathcal{E} \in$ **LES** is a finite set $C \subseteq E_{\mathcal{E}}$ s.t.*

1. *C is* left-closed *: if $e \in C$ and $e' \leq_{\mathcal{E}} e$ then $e' \in C$, and*
2. *C is* conflict-free *: if $e, e' \in C$ then $\neg(e \# e')$.*

We write $Conf(\mathcal{E})$ for the set of all configurations of $\mathcal{E}$.

The empty set of events $\emptyset$ is always a configuration of any $\mathcal{E}$, corresponding to the initial state. Any configuration C of $\mathcal{E}$ is canonically provided with a partial order and a labeling function inherited from $\leq_{\mathcal{E}}$ and $l_{\mathcal{E}}$.

Computation steps simply add new events to the actual configuration. We write $C \xrightarrow{B}_{\mathcal{E}} C'$ when C, C' are two configurations of $\mathcal{E}$ s.t. $C \subseteq C'$ and $B = C' \setminus C$. $C \xrightarrow{B}_{\mathcal{E}} C'$, sometimes simply written $C \rightarrow C'$, is a *transition* in $\mathcal{E}$. As a special case, we write $C \xrightarrow{e}_{\mathcal{E}} C'$ when $C' \setminus C = \{e\}$ and we speak of *atomic transitions*.

We now introduce the notion of computation which is needed to define the mixed-ordering equivalence in Section 2. A computation of a LES is a path in the transition system of configurations. Formally,

Definition 1.4 *Let $\mathcal{E} \in$ **LES**. A* computation *in $\mathcal{E}$ is a sequence $\sigma = e_1 \ldots e_n \in E_{\mathcal{E}}^*$ s.t. $\emptyset \xrightarrow{e_1}_{\mathcal{E}} \{e_1\} \xrightarrow{e_2}_{\mathcal{E}} \{e_1, e_2\} \ldots \xrightarrow{e_n}_{\mathcal{E}} \{e_1, \ldots, e_n\}$. We write C_σ for $\{e_1, \ldots, e_n\}$. σ is* complete *if it cannot be extended (i.e. there is no transition starting from C_σ).*

We write $Comp(\mathcal{E})$ for the set of computations in $\mathcal{E}$, $Exec(\mathcal{E})$ for the set of complete computations, and $\lambda_{\mathcal{E}}$ for the empty computation in $\mathcal{E}$. In Figure 1, $Exec(\mathcal{E}) = \{e_1e_2, e_1e_3, e_3e_1\}$. In this article, $\sigma, \sigma', \rho, \rho', \ldots$ denote computations while $\pi, \pi', \eta, \eta', \ldots$ denote complete computations.

If $\sigma = e_1 \ldots e_n \in Comp(\mathcal{E})$, then $|\sigma| \stackrel{\text{def}}{=} n$ denotes the length of σ. We write $\sigma \leq \sigma'$ when σ is a prefix of σ' (and we say that σ' is an extension of σ). For $0 \leq k \leq |\sigma|$, σ^k ($\stackrel{\text{def}}{=} e_1 \ldots e_k$) denotes the k-th prefix of σ. An expression like $\sigma' = \sigma.\theta$ means that σ' extends σ by the sequence of events θ.

Observe that all configurations are meaningful states in the sense that they can be reached by a computation: for any $C \in Conf(\mathcal{E})$ there is a $\sigma \in Comp(\mathcal{E})$ s.t. $C = C_\sigma$. σ can be understood as state C_σ together with an explicit history of how the state has been reached by a succession of atomic transitions. Observe also that if $\sigma \leq \sigma'$ then $C_\sigma \rightarrow C_{\sigma'}$ so that we sometimes write $\sigma \xrightarrow{\theta}_{\mathcal{E}} \sigma'$ when $\sigma' = \sigma.\theta$.

1.3 Observation of steps

The "observable" part of a sequence of events is often taken to be a "partially ordered multiset" [Pra86], simply called a *pomset*, that is, visible labels together with a causality relation. Formally, a pomset (labeled over *Act*) is the isomorphic class of conflict-free LES's (where $\mathcal{E}$ is *conflict-free* iff $\#_{\mathcal{E}} = \emptyset$). We use letters $p, q, p', \ldots$ to represent pomsets. The size $|p|$ of a pomset is simply the cardinal of the underlying multiset.

Notation 1.5 *Given $a, b \in Act$, we write $a;b$ for the isomorphic class of $(\{e_1, e_2\}, \{(e_1, e_2)\}, \emptyset, \{(e_1, a), (e_2, b)\})$ and $a \mid b$ for the isomorphic class of $(\{e_1, e_2\}, \emptyset, \emptyset, \{(e_1, a), (e_2, b)\})$.*

Given a conflict-free LES $\mathcal{E}$, $pom(\mathcal{E})$ denotes the equivalence class of $\mathcal{E}$ modulo isomorphism. We extend this to arbitrary conflict-free $B \subseteq E_{\mathcal{E}}$ with the induced partial order. Then, if C and C' are configurations s.t. $C \xrightarrow{B}_{\mathcal{E}} C'$, $p \stackrel{\text{def}}{=} pom(B)$ can be understood as the observation of transition $C \xrightarrow{B} C'$ and we write $C \xrightarrow{p} C'$ to denote this. We extend this to (conflict-free) sequences of events, so that we may write $\sigma \xrightarrow{p} \sigma'$ when $\sigma \xrightarrow{\theta} \sigma'$ and $p = pom(\theta)$.

2 Mixed-ordering equivalence

The *mixed-ordering equivalence* [DDM89] is a behavioral equivalence over LES's. It is a variant of bisimulation designed to take causality into account by identifying computations that are *statically* equivalent (they have the same causal structure) as well as *dynamically* equivalent (they have equivalent branching futures).

Notation 2.1 *For $\mathcal{E}, \mathcal{F} \in$ **LES**, $\sigma = e_1 \ldots e_n \in Comp(\mathcal{E})$ and $\rho = e'_1 \ldots e'_n \in Comp(\mathcal{F})$, f_σ^ρ denotes the function with graph $\{(e_i, e'_i) \mid 1 \leq i \leq n\}$.*

Definition 2.2 ***(Mixed-Ordering equivalence)*** *Let $\mathcal{E}, \mathcal{F} \in$ **LES**. A symmetric relation $R \subseteq Comp(\mathcal{E}) \times Comp(\mathcal{F}) \bigcup Comp(\mathcal{F}) \times Comp(\mathcal{E})$ is called a* mixed-ordering equivalence *(or* mo-equivalence *for short) between $\mathcal{E}$ and $\mathcal{F}$, written $R : \mathcal{E} \approx \mathcal{F}$, iff*

1. *initial computations are related : $\lambda_{\mathcal{E}} R \lambda_{\mathcal{F}}$,*
2. *related computations are equivalent : for all $\sigma R \rho$, $f_\sigma^\rho : C_\sigma \mapsto C_\rho$ is an isomorphism, and*
3. *R has the* transfer property *: for all $\sigma R \rho$ and $\sigma \xrightarrow{e} \sigma'$, there exists a $\rho \xrightarrow{e'} \rho'$ s.t. $\sigma' R \rho'$.*

We say that $\mathcal{E}$ and $\mathcal{F}$ are mo-equivalent*, written $\mathcal{E} \approx \mathcal{F}$, iff $R : \mathcal{E} \approx \mathcal{F}$ for some R.*

Point 2 in the above definition is quite strong: $\sigma R \rho$ requires $|\sigma| = |\rho|$, and the transfer property must be satisfied with $l(e) = l(e')$. Also, points 1 and 3 entail that R is total : any computation in $\mathcal{E}$ is related to some computation in $\mathcal{F}$ and vice versa.

Formally R is not an equivalence. However it is easy to check that $\approx$ is an equivalence relation over **LES**. [Ace92] shows that it coincides with history-preserving bisimulation.

Example 2.3 *In Figure 1, $\mathcal{E}$ and $\mathcal{F}$ are not mo-equivalent. To see this assume $R : \mathcal{E} \approx \mathcal{F}$. Then computation $e'_2 \in Comp(\mathcal{F})$ can only be related to $e_1 \in Comp(\mathcal{E})$. But it is no longer possible to find a corresponding equivalent computation to the extension $e_1 e_3$ since the only candidate is $e'_2 e'_4$ but $f_{e_1 e_3}^{e'_2 e'_4}$ is not an isomorphism.*

We conclude this section with two technical lemmas that prove very useful in the next sections. The proofs are left to the reader.

Lemma 2.4 *Assume $R : \mathcal{E} \approx \mathcal{F}$ and $\sigma R \rho$. Then for any $\pi \in Exec(\mathcal{E})$ s.t. $\sigma \leq \pi$, there is a $\eta \in Exec(\mathcal{F})$ s.t. $\rho \leq \eta$ and $\pi R \eta$.*

Lemma 2.5 *Assume $R : \mathcal{E} \approx \mathcal{F}$ and define $\bar{R}$ by $\sigma \bar{R} \rho \overset{def}{\Leftrightarrow} \forall i = 0, \ldots, |\sigma|, \sigma^i R \rho^i$. Then $\bar{R} : \mathcal{E} \approx \mathcal{F}$.*

A corollary is that when $\mathcal{E} \approx \mathcal{F}$, it is possible to pick a prefix-closed mo-equivalence between $\mathcal{E}$ and $\mathcal{F}$ (we say R is prefix-closed whenever R and $\bar{R}$ as defined above coincide.)

3 A back-and-forth modal logic with pomset observations

We aim to define modal characterizations of hp-bisimulation, or equivalently of mixed-ordering equivalence. For this purpose, it is not enough to look at logics which describe properties of the configurations in event structures. Some information about the history of a configuration is required. Because a computation σ can be seen as a configuration C_σ together with a precise history, it seems natural to propose logics which describe properties of computations.

We define L_{bf}, a back-and-forth version of the classical HML logic from [HM85]. Our motivation comes from the characterization of hp-bisimulation as back-and-forth pomset bisimulation [Che92]. L_{bf} contains future and past modalities with pomset observations. Past modalities have already been considered for HML-like logics [HS85, DV90, LPS93]. With additional constructs, past-time modalities were found useful for truly concurrent semantics [DF90, GKP92].

Definition 3.1 ***(Syntax of L_{bf})*** *The formulae of L_{bf} are built according to the following grammar :*

$$\varphi, \psi ::= \top \mid \neg\varphi \mid \varphi \wedge \psi \mid \langle\leftarrow q\rangle\varphi \mid \langle q\rangle\varphi$$

where q is any pomset (over Act).

We use the standard abbreviations $[q]\varphi$, $\bot$, $\varphi \vee \psi$, ... for $\neg\langle q\rangle\neg\varphi$, ... $\langle Act\rangle\varphi$ (resp. $[Act]\varphi$) stands for $\bigvee_{a\in Act}\langle a\rangle\varphi$ (resp. $\neg\langle Act\rangle\neg\varphi$).

A formula of L_{bf} states properties of computations in a given LES. Intuitively speaking, $\langle q\rangle\varphi$ means "now, it is possible to perform pomset q so that φ holds (then)". Formula $\langle\leftarrow q\rangle\varphi$ reads "there exists a time in the past when φ held and s.t. from this time up to now pomset q has been performed". Formally,

Definition 3.2 ***(Semantics of L_{bf})*** *Given $\mathcal{E} \in$ **LES** we define $\models_{\mathcal{E}} \subseteq Comp(\mathcal{E}) \times L_{bf}$ by induction on the structure of formulae :*

- $\sigma \models_{\mathcal{E}} \top$ *always,*
- $\sigma \models_{\mathcal{E}} \neg\varphi$ *iff* $\sigma \not\models_{\mathcal{E}} \varphi$,
- $\sigma \models_{\mathcal{E}} \varphi \wedge \psi$ *iff* $\sigma \models_{\mathcal{E}} \varphi$ *and* $\sigma \models_{\mathcal{E}} \psi$,
- $\sigma \models_{\mathcal{E}} \langle\leftarrow q\rangle\varphi$ *iff there exists* $\sigma' \xrightarrow{q}_{\mathcal{E}} \sigma$ *s.t.* $\sigma' \models_{\mathcal{E}} \varphi$, *and*
- $\sigma \models_{\mathcal{E}} \langle q\rangle\varphi$ *iff there exists* $\sigma \xrightarrow{q}_{\mathcal{E}} \sigma'$ *s.t.* $\sigma' \models_{\mathcal{E}} \varphi$.

We say that $\mathcal{E}$ satisfies φ, written $\mathcal{E} \models \varphi$, whenever $\lambda_{\mathcal{E}} \models_{\mathcal{E}} \varphi$.

Observe that, while future modalities quantify over all possible futures (i.e. extensions of the current computation), there is no real quantification allowed in the semantic clauses for backward modalities. The past is fixed and $\sigma \models \langle\leftarrow q\rangle\varphi$ requires $\sigma^k \models \varphi$ where $k = |\sigma| - |q|$.

Example 3.3 *Using $\mathcal{E}$ and $\mathcal{F}$ as in Figure 1, we have $\mathcal{E} \models \varphi_1$ with $\varphi_1 \stackrel{def}{=} \langle a\rangle\langle b\rangle\langle\leftarrow a;b\rangle\top$ and $\mathcal{E} \models \varphi_2$ with $\varphi_2 \stackrel{def}{=} \langle a\rangle\langle b\rangle\langle\leftarrow a \mid b\rangle\top$. $\mathcal{F}$ also satisfies $\varphi_1 \wedge \varphi_2$.*

A relevant fragment of L_{bf} is L^0_{bf}, where only $\langle a\rangle\varphi$ (for any $a \in Act$) are allowed as forward modalities. φ_1 and φ_2 above are L^0_{bf} formulae but for example $\langle\leftarrow a\rangle\langle\leftarrow b\rangle\langle a \mid b\rangle\top$ is not.

Define $Th_{L_{bf}}(\mathcal{E})$ as $\{\varphi \in L_{bf} \mid \mathcal{E} \models \varphi\}$ and write $\mathcal{E} \equiv_{L_{bf}} \mathcal{F}$ when $Th_{L_{bf}}(\mathcal{E}) = Th_{L_{bf}}(\mathcal{F})$. (Define $Th_{L^0_{bf}}(\mathcal{E})$ and $\mathcal{E} \equiv_{L^0_{bf}} \mathcal{F}$ similarly.) Our first modal characterization result is

Theorem 3.4 *Let $\mathcal{E}$ and $\mathcal{F}$ be LES's. Then $\mathcal{E} \approx \mathcal{F}$ iff $\mathcal{E} \equiv_{L_{bf}} \mathcal{F}$ iff $\mathcal{E} \equiv_{L^0_{bf}} \mathcal{F}$.*

This relies on the following two lemmas and the fact that L^0_{bf} is a fragment of L_{bf}.

Lemma 3.5 *Assume $R : \mathcal{E} \approx \mathcal{F}$ with R prefix-closed. Then $\sigma R \rho$ and $\sigma \models \varphi$ imply $\rho \models \varphi$ for any $\varphi \in L_{bf}$.*

Proof The proof is by induction over the structure of φ. The cases $\top$, $\neg\varphi$, $\varphi \wedge \psi$ are obvious. For the remaining cases, we may assume w.l.o.g. that σ is a computation in $\mathcal{E}$.

- Assume $\sigma \models_{\mathcal{E}} \langle q \rangle \varphi$. Then there exists a transition $\sigma \xrightarrow{\theta}_{\mathcal{E}} \sigma'$ with $pom(\theta) = q$ and $\sigma' \models_{\mathcal{E}} \varphi$. Because R has the transfer property, there exists a transition $\rho \xrightarrow{\theta'}_{\mathcal{F}} \rho'$ s.t. $\sigma' R \rho'$. By induction hypothesis, $\rho' \models_{\mathcal{F}} \varphi$. Also $\sigma' R \rho'$ entails $pom(\theta') = pom(\theta)$ so that $\rho \models_{\mathcal{F}} \langle q \rangle \varphi$.
- Finally assume that $\sigma \models_{\mathcal{E}} \langle \leftarrow q \rangle \varphi$. Then there exists $\sigma' \xrightarrow{q}_{\mathcal{E}} \sigma$ with $\sigma' \models_{\mathcal{E}} \varphi$. σ' is a prefix of σ, say some σ^k, and since R is prefix-closed, we know that $\sigma' R \rho^k$, entailing $\rho^k \models_{\mathcal{F}} \varphi$ by ind. hyp. It remains to see that $\rho^k \xrightarrow{q}_{\mathcal{F}} \rho$ because $pom(C_\rho \setminus C_{\rho^k}) = q$ since f^ρ_σ is an isomorphism.

□

Lemma 3.6 *$\mathcal{E} \equiv_{L^0_{bf}} \mathcal{F}$ implies $\mathcal{E} \approx \mathcal{F}$.*

Proof We write $\sigma \sim \rho$ when σ and ρ satisfy the same L^0_{bf} formulae and show that $\sim : \mathcal{E} \approx \mathcal{F}$. We check the three requirements of Definition 2.2 in turn.

1. $Th_{L^0_{bf}}(\mathcal{E}) = Th_{L^0_{bf}}(\mathcal{F})$ means that $\lambda_{\mathcal{E}} \sim \lambda_{\mathcal{F}}$.
2. Assume $\sigma \sim \rho$ (with σ a computation in $\mathcal{E}$). It is clear that σ and ρ have same length (say, n) because they both satisfy $\langle \leftarrow pom(\sigma) \rangle \top \wedge \langle \leftarrow pom(\rho) \rangle \top$. Similarly, $l_{\mathcal{E}}(\sigma) = l_{\mathcal{F}}(\rho)$. We now show that f^ρ_σ is an isomorphism. Assume it is not. Then there exist pairs (i, j) with $0 < i < j \leq n$ where f^ρ_σ does not respect causality, i.e. s.t. $\sigma(i) \leq_{\mathcal{E}} \sigma(j)) \wedge \rho(i) \not\leq_{\mathcal{F}} \rho(j)$ or *vice versa*. Choose the pair with largest i and smallest $j > i$ and consider $p^j_i \stackrel{\text{def}}{=} pom(\sigma(i)\sigma(i+1)\ldots\sigma(j))$ and $q^j_i \stackrel{\text{def}}{=} pom(\rho(i)\rho(i+1)\ldots\rho(j))$. p^j_i and q^j_i must be distinct pomsets because they do not have the same number of pairs of causally-related events (causality is not respected in exactly one place). Then, writing ψ for $\langle \leftarrow p^n_{j+1} \rangle \langle \leftarrow p^j_i \rangle \top$, we have $\sigma \models \psi$ and $\rho \not\models \psi$, contradicting our hypothesis.
3. It remains to check the transfer property of relation $\sim$: Assume $\sigma R \rho$ and $\sigma \xrightarrow{e}_{\mathcal{E}} \sigma'$. Pick an enumeration $\{\varphi_1, \varphi_2, \ldots\}$ of L^0_{bf} and define $(\psi_i)_{i \in N}$ by

$$\psi_0 \stackrel{\text{def}}{=} \top, \qquad\qquad \psi_{i+1} \stackrel{\text{def}}{=} \begin{cases} \psi_i \wedge \varphi_i & \text{if } \sigma' \models_{\mathcal{E}} \varphi_i, \\ \psi_i \wedge \neg\varphi_i & \text{if } \sigma' \not\models_{\mathcal{E}} \varphi_i. \end{cases}$$

   Write $a \stackrel{\text{def}}{=} l_{\mathcal{E}}(e)$. By construction $\sigma \models_{\mathcal{E}} \langle a \rangle \psi_i$ for all i. Because $\sigma \sim \rho$, $\rho \models_{\mathcal{F}} \langle a \rangle \psi_i$ for all i. Therefore for each i there exists a computation ρ'_i and an event e'_i s.t. $\rho \xrightarrow{e'_i}_{\mathcal{F}} \rho'_i$ with $l_{\mathcal{F}}(e'_i) = a$ and $\rho'_i \models_{\mathcal{F}} \psi_i$.
   By hypothesis, $\mathcal{F}$ is finite. Then there is only a finite number of distinct ρ'_i's. One of them, say ρ'_*, satisfies infinitely many ψ_i's entailing $\sigma' \sim \rho'_*$. Thus $\sim$ has the transfer property.

□

Example 3.7 *The two non mo-equivalent LES's in Figure 1 are distinguished by formula $\langle a \rangle [b] \langle \leftarrow a;b \rangle \top$ which only holds of $\mathcal{F}$.*

4 A path logic with pomsets

We define logic L_P, inspired from [DF90]. It is a path logic (like the CTL^* logic used in [DV90]) in the sense that path modalities (here X and X^{-1}) need not immediately be under the scope of a path quantifier. X^{-1} is a past modality, introduced to make the logic expressive enough for mo-equivalence. Here, and in contrast to the L_{bf} case, pomset observations are propositions and cannot be nested.

Definition 4.1 ***(Syntax of*** L_P***)*** *The formulae of* L_P *are built according to the following grammar :*

$$\varphi, \psi ::= \top \mid q \mid \neg\varphi \mid \varphi \wedge \psi \mid X^{-1}\varphi \mid X\varphi \mid \forall\varphi$$

where q *is any pomset (over Act).*

This comes with the usual abbreviations: $\exists\varphi$ for $\neg\forall\neg\varphi$, ...

As in the original CTL^* logic [EH86], L_P formulae describe properties of states along an execution, i.e. a complete computation, and moreover, because past tense modalities are introduced, it is necessary to keep information on the history of the computation. Therefore L_P formulae are interpreted over models of the form (π, σ), where σ is a computation and π is a complete computation extending σ (the present state). Informally, X^{-1} is the immediate backward modality along past σ and X is the immediate forward modality along π (the selected future). The $\forall$ quantifier provides L_P with branching-time power by quantifying over all possible (complete) futures. Finally, an atomic proposition q holds of (π, σ) if there exists a past time along σ s.t. from this time up to now pomset q has been performed.

Definition 4.2 ***(Semantics of*** L_P***)*** *Given* $\mathcal{E} \in$ ***LES***, $\pi \in Exec(\mathcal{E})$ *and* $\sigma \leq \pi$, *we define* $\models_{\mathcal{E}}$, *the* satisfaction relation, *by induction on the structure of formulae :*

- $(\pi, \sigma) \models_{\mathcal{E}} q$ *iff* σ *can be written as* $\sigma'.\theta$ *with* $pom(\theta) = q$,
- $(\pi, \sigma) \models_{\mathcal{E}} X^{-1}\varphi$ *iff* σ *has the form* $\sigma'.e$ *with* $(\pi, \sigma') \models_{\mathcal{E}} \varphi$,
- $(\pi, \sigma) \models_{\mathcal{E}} X\varphi$ *iff* π *decomposes into* $\pi = \sigma.e.\theta$ *where* $(\pi, \sigma.e) \models_{\mathcal{E}} \varphi$,
- $(\pi, \sigma) \models_{\mathcal{E}} \forall\varphi$ *iff for all* $\pi' \in Exec(\mathcal{E})$ *s.t.* $\sigma \leq \pi'$, $(\pi', \sigma) \models_{\mathcal{E}} \varphi$.

(We omit the obvious clauses for $\top$, $\neg\varphi$ *and* $\varphi \wedge \psi$*.)*

We write $\sigma \models_{\mathcal{E}} \varphi$ *when* $(\pi, \sigma) \models_{\mathcal{E}} \varphi$ *for all complete computations* $\pi \geq \sigma$; *and* $\mathcal{E} \models \varphi$ *iff* $\lambda_{\mathcal{E}} \models_{\mathcal{E}} \varphi$.

We write $\varphi \equiv \psi$, and say that φ and ψ are *semantically equivalent*, when for all π, σ in all $\mathcal{E}$, $(\pi, \sigma) \models \varphi$ iff $(\pi, \sigma) \models \psi$. Here are a few basic properties to be used later :

Proposition 4.3 *For all formulae* $\varphi, \ldots$ *in* L_P,

$$\forall(\varphi_1 \wedge \varphi_2) \equiv \forall\varphi_1 \wedge \forall\varphi_2 \quad (1)$$

$$\forall(\varphi_1 \vee \forall\varphi_2) \equiv \forall\varphi_1 \vee \forall\varphi_2 \quad (2)$$

$$\forall X\varphi \equiv \forall X \forall\varphi \quad (3)$$

$$\neg X\varphi \equiv X\neg\varphi \vee \neg\forall X\top \quad (4)$$

$$X\varphi_1 \vee X\varphi_2 \equiv X(\varphi_1 \vee \varphi_2) \quad (5)$$

Logic L_P as defined above slightly differs (in its atomic propositions) from the original proposal of [DF90], but it is easy to see that any one can be directly encoded into the other.

Define $Th_{L_P}(\mathcal{E})$ and $\mathcal{E} \equiv_{L_P} \mathcal{F}$ in the obvious way. Our second modal characterization result is :

Theorem 4.4 *Let $\mathcal{E}$ and $\mathcal{F}$ be LES. Then $\mathcal{E} \approx \mathcal{F}$ iff $\mathcal{E} \equiv_{L_P} \mathcal{F}$.*

This relies on the following lemmas.

Lemma 4.5 *Assume $R : \mathcal{E} \approx \mathcal{F}$ with R prefix-closed. Then for any complete computations π and η s.t. $\pi R \eta$, for any $k \leq |\pi|$ and $\varphi \in L_P$, $(\pi, \pi^k) \models \varphi$ implies $(\eta, \eta^k) \models \varphi$.*

Proof Assume $\pi R \eta$ and $k \leq |\pi|$. We proceed by induction on the structure of L_P formulae. The cases $\top$, $\neg\varphi$, $\varphi \wedge \psi$ are obvious.

- Assume $(\pi, \pi^k) \models q$. Because f_π^η is an isomorphism, it is easy to see that $(\eta, \eta^k) \models q$ as well.
- The cases $X\varphi$ and $X^{-1}\varphi$ use the induction hypothesis with $k+1$ and $k-1$.
- Finally the case $\forall\varphi$ relies on Lemma 2.4 : assume by way of contradiction that $(\pi, \pi^k) \models \forall\varphi$ while $(\eta, \eta^k) \not\models \forall\varphi$. Then there exists a complete computation η_2 with $\eta_2^k = \eta^k$ s.t. $(\eta_2, \eta^k) \not\models \varphi$. Because R is prefix-closed, $\eta^k R \pi^k$, and Lemma 2.4 entails that there is a π_2 extending π^k s.t. $\pi_2 R \eta_2$ (and π_2 is complete). By ind. hyp., $(\pi_2, \pi^k) \not\models \varphi$ which contradicts the assumption $(\pi, \pi^k) \models \forall\varphi$.

$\square$

The immediate corollary is that $\mathcal{E} \approx \mathcal{F}$ and $\lambda_\mathcal{E} \models_\mathcal{E} \varphi$ entail $\lambda_\mathcal{F} \models_\mathcal{F} \varphi$, establishing the "$\Rightarrow$" direction of Theorem 4.4.

Lemma 4.6 *$\mathcal{E} \equiv_{L_P} \mathcal{F}$ implies $\mathcal{E} \approx \mathcal{F}$.*

A direct proof is possible, mimicking the proof of Lemma 3.6. We omit it here because we shall soon give a simple encoding of L_{bf} in L_P so that $\mathcal{E} \equiv_{L_P} \mathcal{F}$ implies $\mathcal{E} \equiv_{L_{bf}} \mathcal{F}$.

Example 4.7 *Consider $\mathcal{E}$ and $\mathcal{F}$ from Figure 1. Since $\mathcal{E} \not\approx \mathcal{F}$, there must be an L_P formula distinguishing them. Indeed, take $\varphi \stackrel{def}{=} \forall X(a \;\Rightarrow\; \exists X(b \;\wedge\; (a \mid b)))$. Then $\mathcal{E} \models \varphi$ while $\mathcal{F} \not\models \varphi$.*

5 From L_{bf} to L_P

A corollary of Theorems 3.4 and 4.4 is that L_{bf} and L_P have the same distinguishing power, i.e. induce the same equivalence between event structures. But this is a very crude way of comparing their expressive power. It would be much more interesting to know which formulae of a given logic express properties that can be written in the other logic. This is the viewpoint commonly used in the field of temporal logic for concurrent systems [EH86].

In our case there may be a small difficulty with the comparison of L_{bf} and L_P because the two logics do not directly speak of the same thing. While L_{bf} expresses properties of a computation (a state with an history), L_P expresses properties of a computation along a selected future. This is similar to the difficulty inherent in comparing linear-time and branching-time temporal logics [EH86]. This difficulty only appears in one direction because computations without a selected future are simply a special case of computations with a selected future. And indeed, L_{bf} can be seen as a fragment of L_P.

There exists a straightforward translation of L_{bf} into L_P :

Theorem 5.1 *For all $\varphi \in L_{bf}$ there is a φ^* in L_P s.t. for all $\mathcal{E}$ and $\sigma \in Comp(\mathcal{E})$, $\sigma \models_{\mathcal{E}} \varphi$ iff $\sigma \models_{\mathcal{E}} \varphi^*$.*

Proof To any $\varphi \in L_{bf}$ we associate a $\varphi^* \in L_P$ inductively defined by :

$$\top^* \stackrel{\text{def}}{=} \top, \qquad (\neg\varphi)^* \stackrel{\text{def}}{=} \neg(\varphi^*), \qquad (\varphi \wedge \psi)^* \stackrel{\text{def}}{=} \varphi^* \wedge \psi^*,$$

$$(\langle\leftarrow q\rangle\varphi)^* \stackrel{\text{def}}{=} q \wedge X^{-|q|}\varphi^*, \qquad (\langle q\rangle\varphi)^* \stackrel{\text{def}}{=} \exists X^{|q|}(q \wedge \varphi^*).$$

where X^n (resp. X^{-n}) denotes n-nested X (resp. X^{-1}) combinators. $\square$

Observe that the size of φ^* is linear in the size of φ.

Example 5.2 *The L_{bf} formula $\langle a\rangle[b]\langle\leftarrow a;b\rangle\top$ of Example 3.7 translates into L_P formula $\exists X(a \wedge \neg\exists X(b \wedge \neg(a;b \wedge X^{-2}\top)))$.*

6 From L_P to L_{bf}

We now consider the problem of translating L_P formulae into L_{bf}. Because L_P contains linear-time constructs, not every formula can be translated. For example the L_P formula $X a$ is not a property of a computation *per se*, it is a property of the selected future. We can see it as a property of a computation by referring to the definition we gave of "$\sigma \models X a$". Then $\sigma \models X a$ iff $\sigma \models \langle Act\rangle\top \wedge [Act]\langle\leftarrow a\rangle\top$ so that the L_{bf} formula $\langle Act\rangle\top \wedge [Act]\langle\leftarrow a\rangle\top$ is equivalent to the L_P formula $\forall X a$.

This is the correctness criterion we retain :

Theorem 6.1 *For any $\varphi \in L_P$ there is a $\varphi^{\#}$ in L_{bf} s.t. for all $\mathcal{E}$ and $\sigma \in Comp(\mathcal{E})$, $\sigma \models \varphi$ iff $\sigma \models \varphi^{\#}$.*

In essence we only translate the *state formulae* from L_P. Formally, $\varphi \in L_P$ is a *state formula* if the truth of φ at (π, σ) only depends on σ, i.e. for all σ and selected futures π, π', $(\pi, \sigma) \models \varphi \Leftrightarrow (\pi', \sigma) \models \varphi$. Observe that φ is a state formula iff $\varphi \equiv \forall\varphi$ and that boolean combinations of state formulae are state formulae.

There is a syntactical approximation of the semantic notion: say $\varphi \in L_P$ is *branching-time* iff any occurrence of the X modality in φ is immediately under the scope of a $\forall$ or $\exists$ quantifier. For example, $\forall X a$ and $\exists X X^{-1} X^{-1} a$ are branching-time, while $\exists X^{-1} X a$ is not. Write L_P^{br} for the set of branching-time L_P formulae. Observe that all branching-time formulae are state formulae.

The rest of this section is devoted to a proof of Theorem 6.1. We proceed in essentially two steps. First we translate L_P^{br} into L_{bf}, and then translate any L_P state formula into an L_P^{br} formula. (This second step is done by dealing with ever increasing fragments of L_P.)

6.1 Translating L_P branching-time formulae

L_P^{br} is a fragment of L_P that can be directly translated into L_{bf}. To any $\varphi \in L_P^{br}$, we associate a $\varphi^{\#} \in L_{bf}$ inductively defined by

$$\top^{\#} \stackrel{\text{def}}{=} \top, \qquad q^{\#} \stackrel{\text{def}}{=} \langle\leftarrow q\rangle\top, \qquad (\neg\varphi)^{\#} \stackrel{\text{def}}{=} \neg(\varphi^{\#}), \qquad (\varphi \wedge \psi)^{\#} \stackrel{\text{def}}{=} \varphi^{\#} \wedge \psi^{\#},$$

$$(X^{-1}\varphi)^{\#} \stackrel{\text{def}}{=} \langle\leftarrow Act\rangle\varphi^{\#}, \qquad (\forall X\varphi)^{\#} \stackrel{\text{def}}{=} [Act]\varphi^{\#}, \qquad (\exists X\varphi)^{\#} \stackrel{\text{def}}{=} \langle Act\rangle\varphi^{\#},$$

$$(\forall\varphi)^{\#} = (\exists\varphi)^{\#} \stackrel{\text{def}}{=} \varphi^{\#} \text{ when } \varphi \text{ does not have } X \text{ at its root.}$$

Clearly φ and $\varphi^{\#}$ are equivalent. Now a possible strategy to prove Theorem 6.1 is to rewrite any state formula $\varphi \in L_P$ into an equivalent L_P^{br} formula.

6.2 Translating L_P state formulae into L_P^{br} formulae

Here we use separation methods inspired from [Gab87, GPSS80] and adapted to CTL^*-like logics as in [LPS93, LS94]. L_P state formulae, which can be thought as formulae of the form $\forall\varphi$, may contain subformulae where some occurrences of X are not immediately preceded by a $\forall$ operator. In order to translate them into equivalent branching-time formulae, we have to propagate the $\forall$ operator in front of each occurrence of modality X. This is first done for the special case of *future* formulae.

Definition 6.2 ***(Past and future formulae)*** *$\varphi \in L_P$ is a* past *(resp.* future*) formula if it does not contain any occurrence of X (resp. X^{-1}).*

The terminology "past" and "future" only expresses syntactical features of a formula. Observe that any past formula is in L_P^{br}, and then is a state formula.

Lemma 6.3 *Any formula $\forall\psi \in L_P$, where ψ is future, is equivalent to an L_P^{br} formula.*

Proof By induction over the maximal number of nested X modalities in ψ, written $d(\psi)$ in this proof. If $d(\psi) = 0$, then ψ is already in L_P^{br}. For the induction step assume $d(\psi) > 0$. Then with the help of some boolean manipulations (and because it is future) we can rewrite ψ into the general form :

$$\psi \equiv \bigwedge_{i=1,2,\ldots} (\phi_i \vee X\psi_{i,1} \vee X\psi_{i,2} \vee \cdots \vee \neg X\psi'_{i,1} \vee \neg X\psi'_{i,2} \vee \cdots)$$

where the ϕ_i's are L_P^{br} formulae (and thus state formulae) and the $\psi_{i,j}$'s and $\psi'_{i,j}$'s are future. Using (4) and (5), we get the general form :

$$\psi \equiv \bigwedge_{i=1,2,\ldots} (\phi'_i \vee X(\psi_{i,1} \vee \psi_{i,2} \vee \cdots))$$

where the ϕ'_i's are L_P^{br} formulae. Using (2) and (3) we get :

$$\forall\psi \equiv \bigwedge_{i=1,2,\ldots} (\phi'_i \vee \forall X\forall(\psi_{i,1} \vee \psi_{i,2} \vee \cdots))$$

Now, by induction hypothesis, every $\forall(\psi_{i,1} \vee \psi_{i,2} \cdots)$ can be replaced by some equivalent ψ_r in L_P^{br}, which completes the proof. $\square$

Now that we know how to translate future formulae, we use a separation lemma that enlarge the class of formulae we can translate :

Definition 6.4 *$\varphi \in L_P$ is* linear *if it does not contain any occurrence of $\forall$ or $\exists$.*

Lemma 6.5 ***(Separation Lemma)*** *Any linear formula $\varphi \in L_P$ is equivalent to a linear formula of the form $\bigwedge_{i=1,\dots}(\varphi_i^- \vee \varphi_i^+)$, where the φ_i^-'s are past and the φ_i^+'s are future.*

Proof Consider the following rewriting rules which separate past and future modalities in a formula :

$$\begin{array}{lcll}
XX^{-1}\varphi & \longrightarrow & X\top \wedge \varphi & \text{(r1)} \\
X\neg X^{-1}\varphi & \longrightarrow & X\top \wedge \neg\varphi & \text{(r2)} \\
X^{-1}X\varphi & \longrightarrow & X^{-1}\top \wedge \varphi & \text{(r3)} \\
X^{-1}\neg X\varphi & \longrightarrow & X^{-1}\top \wedge \neg\varphi & \text{(r4)}
\end{array}
\qquad
\begin{array}{lcll}
X(\varphi \wedge \psi) & \longrightarrow & X\varphi \wedge X\psi & \text{(r5)} \\
X(\varphi \vee \psi) & \longrightarrow & X\varphi \vee X\psi & \text{(r6)} \\
X^{-1}(\varphi \wedge \psi) & \longrightarrow & X^{-1}\varphi \wedge X^{-1}\psi & \text{(r7)} \\
X^{-1}(\varphi \vee \psi) & \longrightarrow & X^{-1}\varphi \vee X^{-1}\psi & \text{(r8)} \\
\neg(\varphi \wedge \psi) & \longrightarrow & \neg\varphi \vee \neg\psi & \text{(r9)} \\
\neg(\varphi \vee \psi) & \longrightarrow & \neg\varphi \wedge \neg\psi & \text{(r10)} \\
\neg\neg\varphi & \longrightarrow & \varphi & \text{(r11)}
\end{array}$$

- The above rewriting system is clearly correct, that is any rewrite step preserves semantic equivalence of L_P formulae.
- The system is terminating. This can be seen easily by considering the recursive path ordering (see e.g. [Der87]) induced by the following precedence on symbols

$$\top \prec X, X^{-1} \qquad\qquad \neg \prec \wedge, \vee \prec X, X^{-1}$$

- The system is clearly complete, in the sense that any linear formula in which an X modality is in the scope of an X^{-1} modality, or vice versa, can be reduced by one of the above rules, eventually reaching a boolean combination of (linear) past or future formulae.
- Finally these boolean combinations need only be rewritten into conjunctive normal form to conclude the proof.

$\square$

Lemma 6.6 *Any formula $\forall\varphi \in L_P$, where φ is linear, is equivalent to an L_P^{br} formula of the form $\bigwedge_i(\varphi_i^- \vee \psi_i^+)$, where the φ_i^-'s are past and the ψ_i^+'s are future and branching-time.*

Proof Lemma 6.5 enables us to assume w.l.o.g. that formula φ is of the form $\bigwedge_{i=1,\dots}(\varphi_i^- \vee \varphi_i^+)$. Applying (1) yields

$$\forall\varphi \equiv \forall[\bigwedge_{i=1,\dots}(\varphi_i^- \vee \varphi_i^+)] \equiv \bigwedge_{i=1,\dots}\forall(\varphi_i^- \vee \varphi_i^+) \tag{6}$$

Now, because each formula φ_i^- is a past formula, and then a state formula, (2) gives:

$$\forall\varphi \equiv \bigwedge_i(\varphi_i^- \vee \forall\varphi_i^+) \tag{7}$$

Now, invoking Lemma 6.3 on each $\forall\varphi_i^+$ in (7) to replace it by an equivalent L_P^{br} formula ψ_i^+ yields :

$$\forall\varphi \equiv \bigwedge_{i=1,\dots}(\varphi_i^- \vee \psi_i^+)$$

and we are done. $\square$

The last step is to consider formulae of the form $\forall\varphi$, where φ is any L_P formula.

Lemma 6.7 *Any formula $\forall\varphi \in L_P$ is equivalent to an L_P^{br} formula.*

Proof We proceed by induction on φ. Write φ as $\varphi[\phi_1, \ldots, \phi_n]$ where the ϕ_i's are maximal state subformulae. Then, replacing $\phi_1, \ldots, \phi_n$'s by propositional variables $x_1, \ldots, x_n$, we get a linear $\varphi[x_1, \ldots, x_n]$ on which Lemma 6.6 applies, giving

$$\forall\varphi[x_1, \ldots, x_n] \equiv \bigwedge_{i=1,\ldots} (\varphi_i^-[x_1, \ldots, x_n] \vee \psi_i^+[x_1, \ldots, x_n])$$

By induction hypothesis, the ϕ_k's are equivalent to some ϕ'_k's in L_P^{br}. Putting them back in $\varphi[\ldots]$, we get

$$\forall\varphi \equiv \bigwedge_{i=1,\ldots} (\varphi_i^-[\phi'_1, \ldots, \phi'_n] \vee \psi_i^+[\phi'_1, \ldots, \phi'_n])$$

and we are done. $\square$

Conclusion

We have defined L_{bf} and L_P, two modal logics for event structures without silent moves, and have shown they both characterize history-preserving bisimulation. These logics not only allow a better understanding of this basic behavioral equivalence. They also provide direct and natural formal languages in which to express properties, to prove them, and more generally to reason, about truly concurrent systems.

The simplicity of L_{bf} suggests it can be used as an explanation language in automated tools checking for hp-bisimilarity, much like HML is used to explain why two transition systems are not strongly bisimilar [Hil87]. The superior expressivity of L_P suggests it is a more natural language in which to write specifications. This augmented expressivity only exists from a practical viewpoint, as L_P can be translated into L_{bf} and *vice versa*. In effect this translation allows to use any of the two logics in situations where the other is used.

We hope this article and [LPS93] demonstrated that the field of comparative expressivity of modal logics of reactive systems deserves further study. It also deserves to be broadened : as a point in case, we would like to mention that, when translations exist, complexity issues should be given more consideration, in connection with the typical claim that "*in practice* one logic is more expressive than the other".

Acknowledgments

We thank Ferroudja Cherief for her participation in the early stages of this research.

References

[Ace92] L. Aceto. History preserving, causal and mixed-ordering equivalence over stable event structures (note). *Fundamenta Informaticae*, 17(4):319–331, December 1992.

[BC87] G. Boudol and I. Castellani. On the semantics of concurrency: Partial orders and transition systems. In *Proc. CAAP'87, Pisa, LNCS 249*, pages 123–137. Springer-Verlag, March 1987.

[BDKP91] E. Best, R. Devillers, A. Kiehn, and L. Pomello. Concurrent bisimulations in Petri nets. *Acta Informatica*, 28:231–264, 1991.

[BR83] S. D. Brookes and W. C. Rounds. Behavioural equivalence relations induced by programming logics. In *Proc. 10th ICALP, Barcelona, LNCS 154*, pages 97–108. Springer-Verlag, July 1983. Available as CMU Report CMU-CS-83-112.

[Che92] F. Cherief. Investigations of back and forth bisimulations on prime event structures. *Computers and Artificial Intelligence*, 11(5):481–496, 1992.

[CMP87] L. Castellano, G. De Michelis, and L. Pomello. Concurrency vs interleaving: an instructive example. *EATCS Bull.*, 31:12–15, February 1987.

[DD89] Ph. Darondeau and P. Degano. Causal trees. In *Proc. 16th ICALP, Stresa, LNCS 372*, pages 234–248. Springer-Verlag, July 1989.

[DDM89] P. Degano, R. De Nicola, and U. Montanari. Partial orderings descriptions and observations of nondeterministic concurrent processes. In *Linear Time, Branching Time and Partial Order in Logics and Models for Concurrency, Noordwijkerhout, LNCS 354*, pages 438–466. Springer-Verlag, 1989.

[Der87] N. Dershowitz. Termination of rewriting. *J. Symbolic Computation*, 3(1):69–115, February 1987.

[DF90] R. De Nicola and G. L. Ferrari. Observational logics and concurrency models. In *Proc. 10th Conf. Found. of Software Technology and Theor. Comp. Sci., Bangalore, India, LNCS 472*, pages 301–315. Springer-Verlag, December 1990.

[DV90] R. De Nicola and F. Vaandrager. Three logics for branching bisimulation (extended abstract). In *Proc. 5th IEEE Symp. Logic in Computer Science, Philadelphia, PA*, pages 118–129, June 1990.

[EH86] E. A. Emerson and J. Y. Halpern. "Sometimes" and "Not Never" revisited: On branching versus linear time temporal logic. *Journal of the ACM*, 33(1):151–178, 1986.

[Gab87] D. Gabbay. The declarative past and imperative future: Executable temporal logic for interactive systems. In *Proc. Temporal Logic in Specification, Altrincham, UK, LNCS 398*, pages 409–448. Springer-Verlag, April 1987.

[GG89] R. J. van Glabbeek and U. Goltz. Equivalence notions for concurrent systems and refinement of actions. In *Proc. Math. Found. Comp. Sci., Porabka-Kozubnik, LNCS 379*, pages 237–248. Springer-Verlag, August 1989.

[GKP92] U. Goltz, R. Kuiper, and W. Penczek. Propositional temporal logics and equivalences. In *Proc. CONCUR'92, Stony Brook, NY, LNCS 630*, pages 222–236. Springer-Verlag, August 1992.

[GPSS80] D. Gabbay, A. Pnueli, S. Shelah, and J. Stavi. On the temporal analysis of fairness. In *Proc. 7th ACM Symp. Principles of Programming Languages, Las Vegas, Nevada*, pages 163–173, January 1980.

[GR83] U. Goltz and W. Reisig. The non-sequential behaviour of Petri nets. *Information and Control*, 57:125–147, 1983.

[Gra81] J. Grabowski. On partial languages. *Fundamenta Informaticae*, IV(2):427–498, 1981.

[Hil87] M. Hillerström. Verification of CCS processes. M.Sc. Thesis, Aalborg University, 1987.

[HM80] M. Hennessy and R. Milner. On observing nondeterminism and concurrency. In *Proc. 7th ICALP, Noordwijkerhout, LNCS 85*, pages 299–309. Springer-Verlag, July 1980.

[HM85] M. Hennessy and R. Milner. Algebraic laws for nondeterminism and concurrency. *Journal of the ACM*, 32(1):137–161, January 1985.

[HS85] M. Hennessy and C. Stirling. The power of the future perfect in program logics. *Information and Control*, 67:23–52, 1985.

[LPS93] F. Laroussinie, S. Pinchinat, and Ph. Schnoebelen. Translation results for modal logics of reactive systems. In *Proc. AMAST'93, Enschede, NL*, pages 299–310. Springer-Verlag, June 1993.

[LS94] F. Laroussinie and Ph. Schnoebelen. A hierarchy of temporal logics with past. In *Proc. STACS'94, Caen, France, LNCS 775*, pages 47–58. Springer-Verlag, February 1994.

[Mil81] R. Milner. A modal characterisation of observable machine-behaviour. In *Proc. CAAP'81, Genoa, LNCS 112*, pages 25–34. Springer-Verlag, March 1981.

[Pnu85] A. Pnueli. Linear and branching structures in the semantics and logics of reactive systems. In *Proc. 12th ICALP, Nafplion, LNCS 194*, pages 15–32. Springer-Verlag, July 1985.

[Pra86] V. R. Pratt. Modeling concurrency with partial orders. *Int. J. Parallel Programming*, 15(1):33–71, 1986.

[RT88] A. Rabinovich and B. A. Trakhtenbrot. Behavior structures and nets. *Fundamenta Informaticae*, 11(4):357–404, 1988.

[Vaa89] F. Vaandrager. An explicit representation of equivalence classes of the history preserving bisimulation. Unpublished manuscript, CWI-Amsterdam, 1989.

[Vog91] W. Vogler. Bisimulation and action refinement. In *Proc. STACS'91, Hamburg, LNCS 480*, pages 309–321. Springer-Verlag, February 1991.

[Win89] G. Winskel. An introduction to event structures. In *Linear Time, Branching Time and Partial Order in Logics and Models for Concurrency, Noordwijkerhout, LNCS 354*, pages 364–397. Springer-Verlag, 1989.